(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 657 319 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
27.05.2020 Bulletin 2020/22

(21) Application number: **18306551.5**

(22) Date of filing: **22.11.2018**

(51) Int Cl.:
G06F 8/54 *(2018.01)* ***G06F 8/41*** *(2018.01)*
G06F 9/445 *(2018.01)* ***G06F 21/57*** *(2013.01)*

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
KH MA MD TN

(71) Applicant: **Thales Dis France SA**
92190 Meudon (FR)

(72) Inventors:
- **BERTONNIER, Damien**
  92190 MEUDON (FR)
- **REGNAULT, Nicolas**
  92190 MEUDON (FR)
- **MARTIN, Valérie**
  92190 MEUDON (FR)

(74) Representative: **Lotaut, Yacine Diaw**
Thales Dis France SA
Intellectual Property Department
6, rue de la Verrerie
92190 Meudon (FR)

(54) A METHOD FOR GENERATING AN EXECUTABLE FILE FROM A PARENT EXECUTABLE FILE TO PRODUCE A DERIVED CUSTOMER ITEM

(57) The invention relates to a method (P1) for generating an executable file (320), derived from a parent executable file (310) comprising ranges (311) of physical addresses referencing:
- a binary code (312) of at least one core feature (CR);
- a binary code (313) of a set of native features (F) ;
- bytecodes (314) of a set of java features (Pkg);
said method (P1) comprising:
- selecting (E11) at least one native feature (F) from said set of native features to be removed;
- defining (E12) the range (311) of physical addresses where the binary code (313) of said at least one selected native feature (F) is stored;
- selecting (E13) at least one java feature (Pkg) from said set of java features to be relocated;
- relocating (E14) the bytecodes (314) of said at least one selected java feature (Pkg) in said defined range (311) of physical addresses.

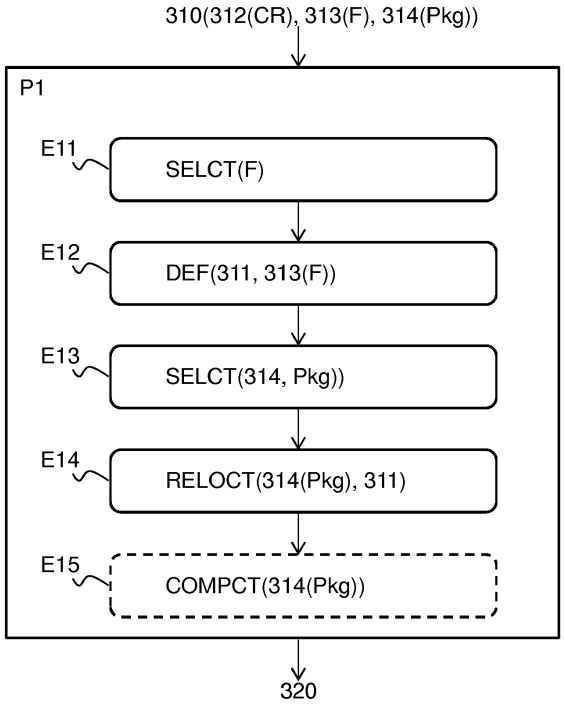

Fig. 1

EP 3 657 319 A1

Description

TECHNICAL FIELD

[0001] The present invention relates to a method for generating an executable file from a parent executable file.

[0002] Such method may be used in a non-limitative example within the domain of certified secure elements.

BACKGROUND OF THE INVENTION

[0003] Secure elements may be used for a wide variety of products and can implement various kinds of technology features. As an example, a secure element can implement security features such as biometry, RSA (Rivest, Shamir, Adleman algorithm) and elliptic curves. An executable file comprises the code of said security features. As an example, a product can be a health card, or a driving license.

[0004] These executable file is generally stored in non volatile memory (NVM). Then, different versions of the secure element can be generated with different subsets of activated features. These different versions are usually designated as derived customer items, and the secure element from which they are derived is designated as a parent customer item. For example, a secure element that includes five different features may be used as a basis to derive several secure elements adapted to different customer requirements by removing one or several features.

[0005] For some secure elements with specific security features, it is sometimes mandatory to certify said secure element, for example using the Common Criteria for Information Technology Security Evaluation standard, ISO 15408. The certification process can be considered as a constraint during the production phase. It can take up to a year for a certification process to be concluded.

[0006] One problem of this prior art is that, when a given parent customer item is certified, the certification process should also be performed on the derived customer items derived from said parent customer item, which implies time and effort required to certify said derived customer items.

SUMMARY OF THE INVENTION

[0007] The following summary of the invention is provided in order to provide a basic understanding of some aspects and features of the invention. This summary is not an extensive overview of the invention and as such it is not intended to particularly identify key or critical elements of the invention or to delineate the scope of the invention. Its sole purpose is to present some concepts of the invention in a simplified form as a prelude to the more detailed description that is presented below.

[0008] It is an object of the invention to provide a method generating an executable file derived from a parent executable file, which permits minimizing the time and effort required by testing and certifying a derived customer item, in which said executable file is embedded.

[0009] To this end, there is provided a method for generating an executable file, derived from a parent executable file, said parent executable file comprising ranges of physical addresses referencing:

- a first binary code of at least one core feature;
- a second binary code of a set of native features;
- bytecodes of a set of java features;

said method comprising:

- selecting at least one native feature from said set of native features to be removed;
- defining the range of physical addresses where the second binary code of said at least one selected native feature is stored;
- selecting at least one java feature from said set of java features to be relocated;
- relocating the bytecodes of said at least one selected java feature in said defined range of physical addresses.

[0010] Said executable file is designated as derived executable file.

[0011] As we will see in further details, as the execution file derived from the parent executable file, is only a modified parent executable file where one or a plurality of native features have been removed, the physical addresses of the first binary code of the at least one core feature and the second binary code of the remaining native features of the set of native features have not changed, and thus the first binary code and the second binary code have not been modified. Therefore, the certification of a secure element in which said parent executable file is embedded may be automatically applied to another secure element in which said derived executable file is embedded, without having another certification process to be performed for said another secure element.

[0012] According to non-limitative embodiments of the invention, the method in accordance with the invention further comprises the following characteristics.

[0013] In a non-limitative embodiment, said at least one core feature and said set of native features comprise a native code, and said set of java features comprises the java code of at least one java package.

[0014] In a non-limitative embodiment, said first binary code, said second binary code and said bytecodes are generated by compiling a source code comprising said at least one core feature, said set of native features and said set of java features.

[0015] In a non-limitative embodiment, said at least one core feature is not removable.

[0016] In a non-limitative embodiment, the relocating is performed according to the bytecodes' size of said at least one selected java feature and the defined range of

physical addresses' size.

[0017] In a non-limitative embodiment,

- said set of native of features comprises a plurality of native features;
- said set of java features comprises a plurality of java features.

[0018] In a non-limitative embodiment, a plurality of native features is selected to be removed.

[0019] In a non-limitative embodiment, the bytecodes of a plurality of java features is relocated.

[0020] In a non-limitative embodiment, said method further comprises compacting the bytecodes of the java feature(s) which are not relocated.

[0021] In a non-limitative embodiment, the range(s) of physical addresses freed by the bytecodes compacting is reserved for user data.

[0022] In a non-limitative embodiment, the parent executable file is embedded in a non-volatile memory of a secure element designated as a parent customer item.

[0023] In addition, there is provided an executable file, wherein said executable file is generated according to the method characterized according to any characteristics above-mentioned.

[0024] In addition, there is provided a method for producing a secure element designated as a derived customer item, said secure element comprising a non-volatile memory, wherein said method comprises the loading of an executable file generated according to the method of any above-mentioned characteristics, in said non-volatile memory of said secure element.

[0025] In a non-limitative embodiment, said non-volatile memory is a flash memory.

[0026] In a non-limitative embodiment, said secure element is an integrated circuit card.

[0027] In addition, there is provided a secure element designated as a derived customer item, which is produced by the method above-mentioned.

[0028] In a non-limitative embodiment, said secure element is an integrated circuit card.

[0029] To achieve those and other advantages, and in accordance with the purpose of the invention as embodied and broadly described, the invention proposes a method for generating an executable file derived from a parent executable file, said parent executable file comprising ranges of physical addresses referencing:

- a second binary code of at least a native feature which is removable from the parent executable file;
- bytecodes of at least a java feature;

wherein during the generation of the derived executable file:

- selecting at least one native feature from said set of native features to be removed;
- defining the range of physical addresses where the second binary code of said at least one selected native feature is stored in the parent executable file;
- selecting at least one java feature from said set of java features to be relocated;
- in the generated derived executable file, relocating the bytecodes of said at least one selected java feature in said defined range of physical addresses according to a predefined optimization relocation algorithm.

[0030] In a various method of the present invention, the ranges of physical addresses of said parent executable file reference a first binary code of at least one core feature which is not removable from the parent executable file, and wherein said at least one core feature is not removable from the derived executable file and wherein the first binary code is stored in the derived executable file in the same range of physical addresses than in the parent executable file.

[0031] In a various method of the present invention, the relocating process is performed according to the size of the bytecodes to be relocated and the size of the defined range of the physical addresses.

[0032] In a various method of the present invention, the bytecodes of the java feature(s) which are not relocated is compacted.

[0033] In a various method of the present invention, the range(s) of physical addresses freed by the compacted bytecodes (314) is reserved for user data.

[0034] In a various method of the present invention, the parent executable file is loaded in a non-volatile memory.

[0035] In a various method of the present invention, the generated derived executable file is loaded in a non-volatile memory of a secure element designated as a derived customer item.

[0036] In a various method of the present invention, the non-volatile memory is a flash memory.

[0037] In a various method of the present invention, a native feature F is selected to be removed when there is no dependency between said native feature and another native feature.

[0038] In a various method of the present invention, a java feature is selected for relocation when its bytecodes can be shifted in memory in any range of physical addresses without considering the other bytecodes of the other java features.

[0039] In a various method of the present invention, the native feature to be removed is overwritten by the relocation of the bytecodes or deleted before the relocation process.

BRIEF DESCRIPTION OF THE DRAWINGS

[0040] Some embodiments of methods and/or apparatus in accordance with embodiments of the present invention are now described, by way of example only, and with reference to the accompanying drawings, in which:

- Fig. 1 is an organizational chart of a method for generating an executable file derived from a parent executable file, according to a non-limitative embodiment of the invention;
- Fig. 2 illustrates a diagram of a parent executable file from which an executable file is derived according to the method of Fig. 1, said parent executable file comprising a first binary code of a core feature, a second binary code of a set of native features, and bytecodes of java features, in a non limitative embodiment;
- Fig. 3 illustrates a diagram of a source code from which the parent executable file of Fig. 2 is generated;
- Fig. 4 illustrates a diagram of the parent executable file of Fig. 2, wherein the binary code corresponding to two native features has been removed, according to the method of Fig. 1, in a non-limitative embodiment;
- Fig. 5 illustrates a diagram of the parent executable file of Fig. 4, wherein the bytecodes of five java features have been relocated, according to the method of Fig. 1, in a non-limitative embodiment;
- Fig. 6 illustrates a diagram of the parent executable file of Fig. 5, wherein the bytecodes of some java features which have not been relocated have been compacted for producing a derived executable file, according to the method of Fig. 1, in a non-limitative embodiment;
- Fig. 7 illustrates an architecture of a parent secure element in which said parent executable file of Fig. 2 is embedded, in a non limitative embodiment;
- Fig. 8 illustrates an architecture of a derived secure element in which said derived executable file of Fig. 6 is embedded, in a non limitative embodiment;
- Fig. 9a illustrates a non-volatile memory of the parent secure element of Fig. 7, in a non-limitative embodiment;
- Fig. 9b illustrates a non-volatile memory of the parent secure element of Fig. 8, in a non-limitative embodiment; and
- Fig. 10 is an organizational chart of a method for producing a secure element in which said derived executable file of Fig. 8 is embedded, according to a non-limitative embodiment of the invention.

DESCRIPTION OF EMBODIMENTS OF THE INVENTION

[0041] Throughout the figures, the same reference numerals and characters, unless otherwise stated, are used to denote like features, elements, components or portions of the illustrated embodiments.

[0042] In the following description, numerous specific details are set forth.

[0043] However, it is understood that well-known circuits, structures, techniques functions or constructions by the man skilled in the art are not described in detail since they would obscure the invention in unnecessary detail.

[0044] The present invention relates to a method P1 for generating an executable file 320 derived from a parent executable file 310.

[0045] In a non-limitative embodiment, the parent executable file 310 is in the HEX format. As will be described in the following, one or several portions of the parent execution file 310 corresponding to some native feature(s) F to be removed are replaced by bytecodes 324 of java features Pkg.

[0046] As illustrated in Fig. 2, the parent executable file 310 comprises ranges 311 of physical addresses referencing:

- a first binary code 312 of at least one core feature CR;
- a second binary code 313 of a set of native features F;
- bytecodes 314 of a set of java features Pkg.

[0047] In the following, a set of native features or a native feature will be referenced F. In the following, a set of java features or a java feature will be referenced Pkg.

[0048] In a non-limitative embodiment, the set of native features F comprises one or a plurality of native features. In the non-limitative illustrated example in Fig. 2 and Fig. 3, the parent executable file 310 comprises the second binary code 313-1 to 313-5 of five native features F1 to F5.

[0049] In non-limitative examples:

- a first native feature F1 implements RSA algorithm;
- a second native feature F2 implements an error correcting codes EXX algorithm.
- a third native feature F3 implements RSA on-board key generation OBKG algorithm.

[0050] In a non-limitative embodiment, the set of java features Pkg comprises one or a plurality of java features Pkg. In the non-limitative illustrated example in Fig. 2 and Fig. 3, the parent executable file 310 comprises the bytecodes 314-1 to 314-8 of eight java features Pkg-1 to Pkg-8.

[0051] The parent executable file 310 is obtained by:

- compiling a source code 300 illustrated in Fig. 3 comprising said at least one core feature CR, the set of native features F and the set of java features Pkg, said compiling resulting in said first binary code 312, said second binary code 313 and said bytecodes 314. Hence, the first binary code 312 is the compiled code of the at least one core feature CR, the second binary code 313 is the compiled code of the set of native features F and the bytecodes 314 are the compiled code of the set of java features Pkg. In a non-limitative embodiment, the compiling generates an object file .obj comprising said first binary code 312 and said second binary code 313, and a .jca file com-

prising said bytecodes 314;

- linking the object file .obj and the .jca file with said ranges 311 of physical addresses resulting in the parent executable file 310. The parent executable file 310 comprises the ranges 311 of physical addresses illustrated in Fig. 2 where the codes (first binary code 312, second binary code 313, and bytecodes 314) are positioned. In a non-limitative embodiment, the ranges 311 of physical addresses are contiguous. Hence, after the linking, different ranges 311 of physical addresses are related to the different codes.

[0052] It is to be noted that the ranges 311 of physical addresses are those of the non-volatile memory NVM described in the following, where the parent executable file 310 is loaded. It is further to be noted that the first range 311-1 of physical addresses depends on the location where the parent executable file 310 is loaded in the non-volatile memory NVM.

[0053] In this description, a distinction is made between core features CR which are mandatory components for the source code 300, and non-core features which are optional components that may be removed from the parent executable file 310 to obtain a derived executable file 320 described in the following. Unless explicitly mentioned, a native feature refers to a non-core feature.

[0054] The core features CR are implementing the minimum functionalities to be embedded in a secure element 11 which is a parent secure element, also designated as a parent customer item or in a secure element 12 which is derived from said parent customer item and designated as a derived secure element or as a derived customer item.

[0055] In a non-limitative embodiment, said at least one core feature CR and the set of native features F comprise a native code, and the set of java features Pkg comprises the java code of at least one java package. A java package Pkg is a library.

[0056] Native code refers to programming code that is configured to run on a specific processor. In a non-limitative example, the native code is in C code or assembly code.

[0057] Bytecodes of the java code are interpreted by a Java Virtual Machine into code that will run on computer hardware.

[0058] The parent executable file 310 is a computer program product which is embedded in a secure element 11 designated as parent customer item. The parent executable file 310 generated for the parent customer item 11 can be used as a basis to produce executable files, designated as derived executable files, for derived customer items 12.

[0059] In a non-limitative element, said secure element 11 is an integrated circuit card ICC also designated as ICC card in the following. The ICC card may be contact or contactless.

[0060] In non-limitative embodiments, the ICC card is a smart device, a soldered element, a M2M module, an eSE (embedded secure element), a micro-SD etc.

[0061] In non-limitative examples, the ICC card is a banking card, such as an EMV card, an Electronic Identity Card, a health card, a driving license, a passport, a privacy card, a financial service card, an access card etc.

[0062] It is to be noted that secure elements are able to control the access to the data they contain and to authorize or not the use of data by other machines. Secure elements may also provide computation services based on cryptographic components. In general, secure elements have limited computing resources and are intended to be connected to a host machine. Secure elements may be removable or fixed to a host device.

[0063] Fig. 7 is a non-limitative embodiment of an architecture of said secure element 11. It comprises:

- a central processing unit (CPU) 110;
- a non-volatile memory (NVM) 112 ;
- a random access memory (RAM) 113;
- a communication interface (I/O) 115 for receiving input and placing output to a computer network, e.g., the Internet, to which the secure element 11 may be connected, either directly or via intermediary devices, such as a host computer. These various components are connected to one another, for example, by a bus 116 ;
- a Java Virtual Machine (JVM) 117.

[0064] In a non-limitative embodiment illustrated in Fig. 7, the non-volatile memory NVM is a FLASH memory.

[0065] In a non-limitative embodiment, the parent executable file 310, is embedded in the secure element 11, designated as parent customer item, and is stored in the non-volatile memory NVM. In a non-variant of embodiment, it is stored in the FLASH memory. In other non-variant of embodiment, it is stored in other types of non-volatile memory.

[0066] A non-limitative example of the non-volatile memory NVM where the parent executable file 310 is loaded is illustrated in Fig. 9a. As illustrated, it comprises the parent executable file 310 within the ranges 311-1 to 311-14 and a memory space for the user data UD in the range 311-10 of physical addresses.

[0067] In a non-limitative embodiment, the non-volatile memory NVM comprises the Java Virtual Machine JVM.

[0068] During operation, the CPU 110 executes the instructions of the different codes (first binary code 312, second binary code 313) stored in the stored in the non-volatile memory NVM, and the Java Virtual Machine JVM interprets the bytecodes 314 stored in the non-volatile memory NVM.

[0069] The executable file 320, designated as derived executable file, which is derived from the parent executable file 310, comprises:

- the first binary code 312 of said at least one core

feature CR;

- a second binary code 313 of a sub-set of the set of native features F of the parent executable file 310;
- bytecodes 314 of said set of java features Pkg of the parent executable file 310.

[0070] Such an executable file 320 is illustrated in Fig. 6, in a non-limitative embodiment. It results from a modification of the parent executable file 310 where at least one native feature F has been removed, that is to say where the corresponding second binary code 313 has been replaced.

[0071] The method P1 for generating such an executable file 320 derived from the parent executable file 310 is illustrated in Fig. 1. It comprises the following steps:

In step E11), illustrated SELCT(F), at least one native feature F from the set of native features is selected to be removed.

[0072] In the non-limitative embodiment, a plurality of native features F is selected to be removed. In the non-limitative example illustrated in Fig. 4, the native features F2 and F4 are selected. In Fig. 4 the native features F2 and F4 are circled to show their selection.

[0073] It is to be noted that the core features CR are not removable. Therefore, the first binary code 312 will be stored in the derived executable file 320 in the same range 321 of physical addresses than in the parent executable file 310.

[0074] It is to be noted that if there is a dependency between a native feature F and another native feature F, it can't be selected to be removed.The word "dependency" should be understood as a functional link between two native features F. These functional links are introduced by instructions in the source code 300, for example "call" or "jump" instructions.

[0075] **In step E12)**, illustrated DEF(311, 313(F)), the range 311 of physical addresses where the second binary code 313 of said at least one selected native feature F is stored, is defined.

[0076] In the non-limitative example illustrated in Fig. 2 and Fig. 3, the second binary codes 313-2 and 313-4 of the native features F2 and F4 are stored within the ranges 311-11 and 311-13 of physical addresses. It is to be reminded that the storage in said ranges 321-2 and 321-4 has been defined from the linking process.

[0077] **In step E13)**, illustrated SELCT(314, Pkg)), at least one java feature Pkg from the set of java features to be relocated is selected.

[0078] In a non-limitative embodiment, the bytecodes 314 of a plurality of java features Pkg is selected to be relocated. In a non-limitative example, the java features Pkg-3, Pkg-4, Pkg-6, and Pkg-8 are selected.

[0079] It is to be noted that a java feature Pkg which can be relocated has no dependency with other java feature Pkg. The word "dependency" should be understood as a functional link between two java features Pkg. These functional links are introduced by calling in the source code 300, the name of the java feature Pkg.

[0080] Hence, when there are a plurality of java features Pkg, the java features Pkg which are relocatable are independent from each other, which means that their bytecodes 314 can be shifted in memory in any range 311 of physical addresses without considering the other bytecodes 314 of the other java features Pkg.

[0081] **In step E14)**, illustrated RELOCT(314(Pkg), 311), the bytecodes 314 of said at least one selected java feature Pkg are relocated in said defined range 311 of physical addresses.

[0082] In a non-limitative embodiment, the bytecodes 314 of a plurality of java features Pkg are relocated. In the non-limitative example illustrated in Fig. 5, the bytecodes 314-3, 314-4, 314-6, and 314-8 respectively of the java features pkg3, pkg4, pgk6 and Pkg8 are relocated.

[0083] In a non-limitative embodiment, the relocating is performed according to the bytecodes' size of said at least one selected java feature Pkg and the defined range 311 of physical addresses' size.

[0084] In a non-limitative embodiment, the relocating is performed according to the Best Fit algorithm well-known by the man skilled in the art. It permits to optimize the relocating.

[0085] Hence, in the non-limitative example illustrated in Fig. 5, according to the size of the range 311-11 of physical addresses and the range 311-13 of physical addresses, and to the size of the bytecodes 314-3, 314-4, 314-6, and 314-8, the bytecodes 314-3, 314-4 fit in the range 311-13 of physical addresses and therefore are relocated in said range 311-13, and the bytecodes 314-6, and 314-8 fit in the range 311-11 of physical addresses and therefore are relocated in said range 311-11.

[0086] It is to be noted that after the relocating, there may be some part of the second binary code 313 of the native feature(s) F (which has been selected to be removed), which has not been overwritten by the relocation of the bytecodes 314 and which is left in the defined range 311 of physical addresses. This part of second binary code 313 is a dead code because it won't be called anymore.

[0087] In a non-limitative embodiment, before the relocating, the second binary code 313 of said at least one selected native feature F may be deleted from said parent executable file 310. In another non-limitative embodiment, after the relocating, the part of the second binary code 313 which is left may be deleted from said parent executable file 310, part which has not been overwritten by the relocating of the bytecodes 314. The deleting may be performed by filling with bits 0 or 1 the range 311 of physical addresses corresponding to said part left.

[0088] **In step E15)** illustrated COMPACT(314(Pkg), when the bytecodes 314 of the selected java feature(s) Pkg have been relocated, in a non-limitative embodiment, the method P1 further comprises compacting the bytecodes 314 of the java features Pkg which are not relocated. As this step is non mandatory, it is illustrated in dotted line in Fig. 1.

[0089] It permits to free some memory space.

[0090] In a non-limitative embodiment, the range(s) 311 of physical addresses freed by the bytecodes 314 compacting is reserved for user data UD in the non-volatile memory NVM where the derived executable file 320 is loaded, that is to say data which are specific to the user of the derived customer item 12. In non-limitative examples, user data UD comprise a picture, a signature, the name, address, age of the user etc.

[0091] In a non-limitative example illustrated in Fig. 6, when the bytecodes 314-3, 314-4, 314-6, and 314-8 of the java features Pkg3, Pkg4, Pkg6, and Pkg8 have been relocated, it has made:

- some memory space available in the ranges 311-3 and 311-4 of physical addresses between the bytecodes 314-2 and 314-5 of the remaining java features Pkg2 and Pkg5;
- some memory space available in the range 311-6 of physical addresses;
- some memory space available in the 311-8 of physical addresses;

[0092] Fig. 6 illustrated the result of the compacting step which produces the derived executable file 320 which is loaded in a secure element for producing a derived customer item. As illustrated, the derived executable file 320 is smaller than the parent executable file 310 due to the compacting.

[0093] It is to be noted that if the method P1 comprises no compacting step, the derived executable file 320 generated is the one illustrated in Fig. 5, that is to say it results from the step E14.

[0094] By compacting the bytecodes 314 of the remaining java features Pkg, it increases the memory space available for the user data UD in the non-volatile memory NVM in which the derived executable file 310 is loaded.

[0095] A non-limitative example of the non-volatile memory NVM where the derived executable file 320 is loaded is illustrated in Fig. 9b. As illustrated, it comprises the derived executable file 320, some user data UD, and some added memory space BPU where more user data UD can be added.

[0096] This memory space brings an added value to the derived customer item 12 and this can be monetized. For a predetermined size of non-volatile memory NVM, removing unnecessary native features F allows to maximize the size of the user data memory by adding unused memory to it and therefore increase the revenue generated while selling the derived customer item 12.

[0097] A method P2 for producing a secure element 12, designated as derived customer item, said secure element 12 comprising a non-volatile memory NVM, is illustrated in Fig. 10. Said method P2 comprises the loading (step E21 illustrated LD(320, NVM)) of said executable file 320 in said non-volatile memory NVM of said secure element 12. Thus, a derived customer item 12 is produced from the parent customer item 11 in which the parent executable file 310 is embedded.

[0098] The description made for the secure element 11 is applied for the secure element 12.

[0099] In a non-limitative embodiment, as the same manner than the secure element 11, the secure element 12 is an integrated circuit card ICC.

[0100] Fig. 8 is a non-limitative embodiment of architecture of said secure element 12. It comprises:

- a central processing unit (CPU) 120;
- a non-volatile memory (NVM) 122 ;
- a random access memory (RAM) 123;
- a communication interface (I/O) 125 for receiving input and placing output to a computer network, e.g., the Internet, to which the secure element 11 may be connected, either directly or via intermediary devices, such as a host computer. These various components are connected to one another, for example, by a bus 126 ;
- a Java Virtual Machine (JVM) 127.

[0101] In a non-limitative embodiment illustrated in Fig. 8, the non-volatile memory NVM is a FLASH memory.

[0102] In a non-limitative embodiment, the derived executable file 320, is embedded in the secure element 12 and is stored in the non-volatile memory NVM. In a non-variant of embodiment, it is stored in the FLASH memory. In other non-variant of embodiments, it is stored other types of non-volatile memory.

[0103] In a non-limitative embodiment, the non-volatile memory NVM comprises the Java Virtual Machine JVM.

[0104] During operation, the CPU 120 executes the instructions of the different codes (first binary code 312, second binary code 313) stored in the stored in the non-volatile memory NVM, and the Java Virtual Machine JVM interprets the bytecodes 314 stored in the non-volatile memory NVM.

[0105] Hence, with the derived executable file 320, a secure element 12, designated as derived customer item, is derived from the parent customer item 11.

[0106] Hence, a plurality of derived customer items 12 comprising different combinations of native features F can be produced in this manner.

[0107] Hence, thank to the generation of the derived executable file 320 described, the compilation of the source code 300 and the linking of the object file .obj and the .jca file is performed only one time and for all the customer items, whether it is a parent customer item 11 or a derived customer item 12.

[0108] If the parent customer item 11 embedding this parent executable file 310 is tested and certified, all derived customer items 12 derived from this parent customer item 11 will be considered as already tested and certified.

[0109] During the certification process, only the features (java, native and core) of the derived customer item 12 are checked and compared to the features (native and core) of the parent customer item 11. Therefore, if some

native features F of the parent customer item 11 are removed from the derived customer item 12, there is no consequence for the certification process.

[0110] For the certification process, the native code (of the core feature(s) CR and of the native feature(s) F) checked must be the same, and its related ranges 311 of physical addresses must be the same. It is to be noted that even if only one related range 311 of physical addresses is changed, it implies that the native code is modified, as the call from one native feature F for example will be modified when it calls another native feature F which related range 311 of physical addresses (where its second binary code 313 is positioned) has been modified.

[0111] For the certification process, the java code (of the java feature(s) Pkg) checked must also be the same. But, if it is relocated, there is no consequence regarding the java code. It is not modified, as the dependency between different java features Pkg is implemented according to the calling of the name of the java feature Pkg. The Java Virtual Machine JVM interprets the dependency according to a table of the ranges 311 of physical addresses which are related to the different java features Pkg. When a java feature Pkg is relocated, this table is updated with the new corresponding ranges 311 of physical addresses.

[0112] It is to be understood that the present invention is not limited to the aforementioned embodiments.

[0113] Hence, some embodiments of the invention may comprise one or a plurality of the following advantages:

- it allows producing different versions of a secure element, based on the same hardware platform, but taking into account the needs of different customers or applications. The production process is fast, as there is no re-compilation of re-linking required for the derived customer items 12. Furthermore, it is not anymore required to re-test and re-certify the derived customer items 12 if the parent customer item 11 is already certified;
- it avoids performing re-compilation and re-linking for derived customer items 12 at product generation time or on the field for personalizing products;
- it permits to produce some derived customer items 12 when the secure element doesn't comprise any memory management unit designated as MMU, such MMU allowing the linking of virtual addresses to an object file and the mapping of said virtual addresses with ranges of physical addresses;
- memory manufacturers are sometimes selling their product (the non-volatile memory NVM) using pay-per-use billing (also referred as bill-per-use, BPU). This means that the selling price of the memory will depend on the amount of the memory that is really used by a secure element. Therefore, by removing unnecessary native features F, the unused memory space is optimized and the production cost of a derived customer item 12 is reduced.

Claims

1. A method (P1) for generating an executable file (320) derived from a parent executable file (310), said parent executable file (310) comprising ranges (311) of physical addresses referencing:

   - a second binary code (313) of at least a native feature (F) which is removable from the parent executable file;
   - bytecodes (314) of at least a java feature (Pkg);

   wherein during the generation of the derived executable file:

   - selecting (E11) at least one native feature (F) from said set of native features (F) to be removed;
   - defining (E12) the range (311) of physical addresses where the second binary code (313) of said at least one selected native feature (F) is stored in the parent executable file;
   - selecting (E13) at least one java feature (Pkg) from said set of java features to be relocated;
   - in the generated derived executable file, relocating (E14) the bytecodes (314) of said at least one selected java feature (Pkg) in said defined range (311) of physical addresses according to a predefined optimization relocation algorithm.

2. A method (P1) according to the previous claim, wherein ranges (311) of physical addresses of said parent executable file (310) reference a first binary code (312) of at least one core feature (CR) which is not removable from the parent executable file, and wherein said at least one core feature (CR) is not removable from the derived executable file and wherein the first binary code is stored in the derived executable file in the same range of physical addresses than in the parent executable file.

3. A method (P1) according to any previous claims, wherein the relocating process is performed according to the size of the bytecodes to be relocated and the size of the defined range (321) of the physical addresses.

4. A method (P1) according to any previous claims, further comprising the step of compacting (E15) the bytecodes (314) of the java feature(s) (Pkg) which are not relocated.

5. A method (P1) according to the previous claim, wherein the range(s) (311) of physical addresses freed by the compacted bytecodes (314) is reserved

for user data (UD).

6. A method (P1) according to any previous claims, wherein the parent executable file (310) is loaded in a non-volatile memory (112).

7. A method (P1) according to any previous claims, wherein said method comprising the loading of the derived executable file (320) generated according to the method of any previous claims 1 to 10, in a non-volatile memory (122) of a secure element (12) designated as a derived customer item.

8. A method (P2) according to the previous claim 6 and 7, wherein said non-volatile memory (122) is a flash memory.

9. A method (P1) according to any previous, wherein the parent executable file (310) is loaded in a secure element (11) designated as a parent customer item

10. A method (P1) according to claim 7 or claim 9, wherein said secure element is an integrated circuit card (ICC).

11. A method (P1) according to any previous claims, wherein a native feature is selected to be removed when there is no dependency between said native feature and another native feature.

12. A method (P1) according to any previous claims, wherein a java feature is selected for relocation when its bytecodes can be shifted in memory in any range of physical addresses without considering the other bytecodes of the other java features.

13. A method (P1) according to any previous claims, wherein the native feature to be removed is overwritten by the relocation of the bytecodes or deleted before the relocation process.

14. A method (P1) according to any previous claims, wherein said core feature (CR) and said native feature (F) comprise a native code, and said set of java features (Pkg) comprises the java code of at least one java package.

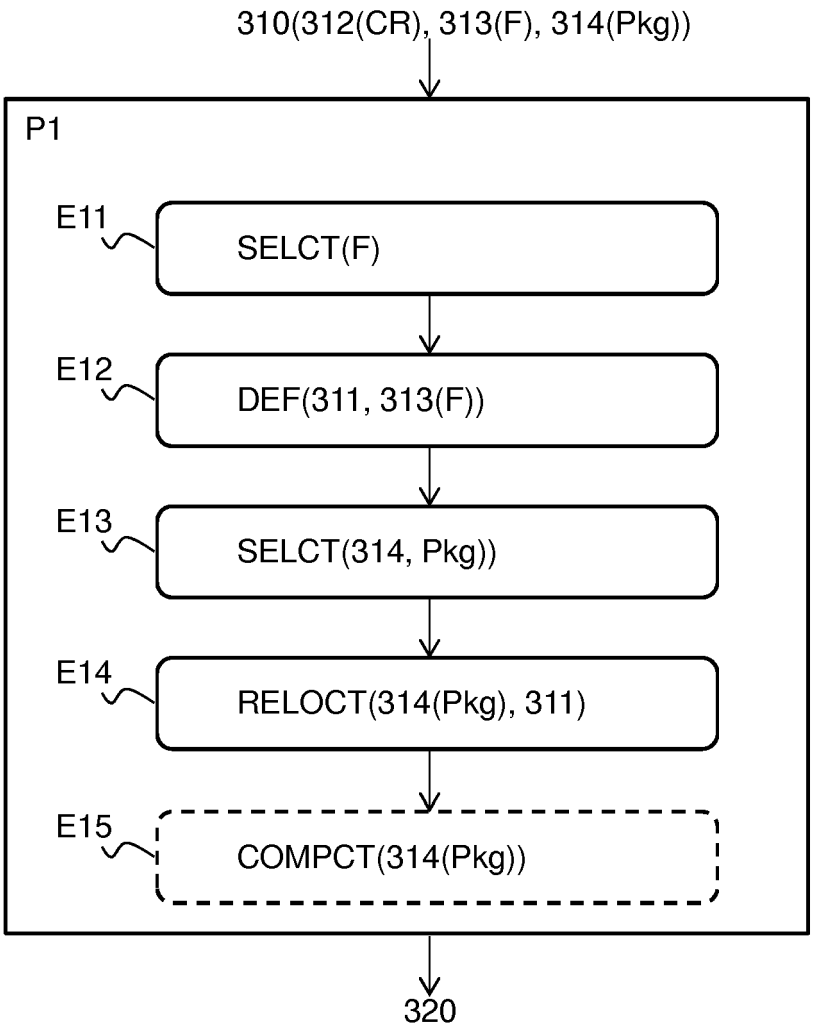
310(312(CR), 313(F), 314(Pkg))
P1
E11
SELCT(F)
E12
DEF(311, 313(F))
E13
SELCT(314, Pkg))
E14
RELOCT(314(Pkg), 311)
E15
COMPCT(314(Pkg))
320

Fig. 1

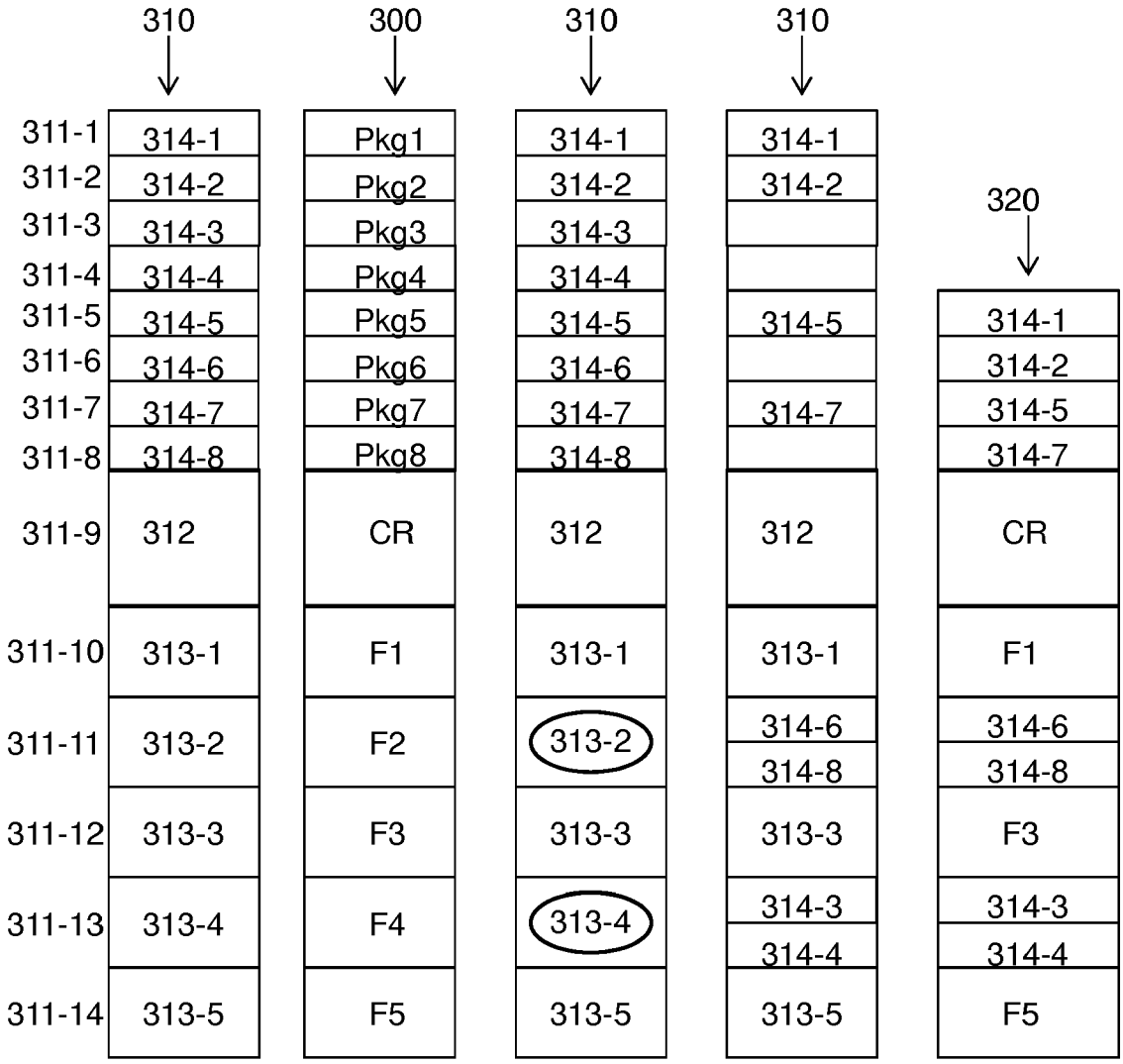

Fig. 2 **Fig. 3** **Fig. 4** **Fig. 5** **Fig. 6**

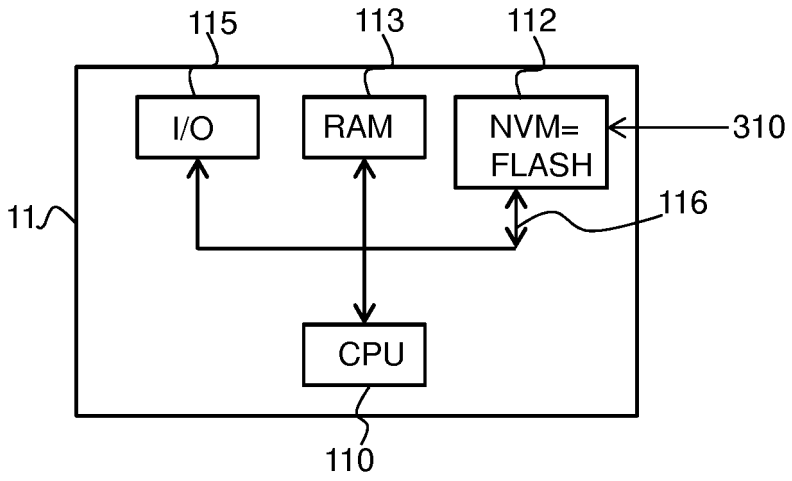

Fig. 7

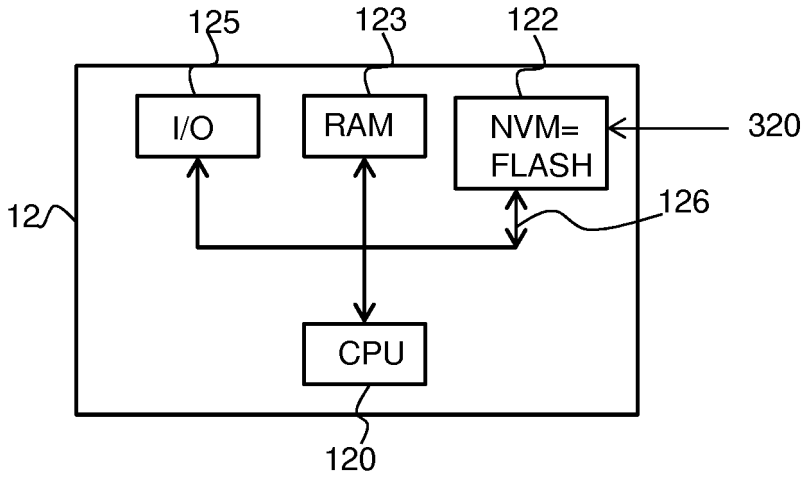

Fig. 8

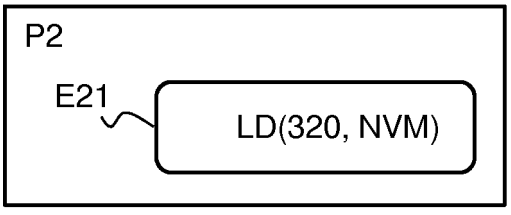

Fig. 10

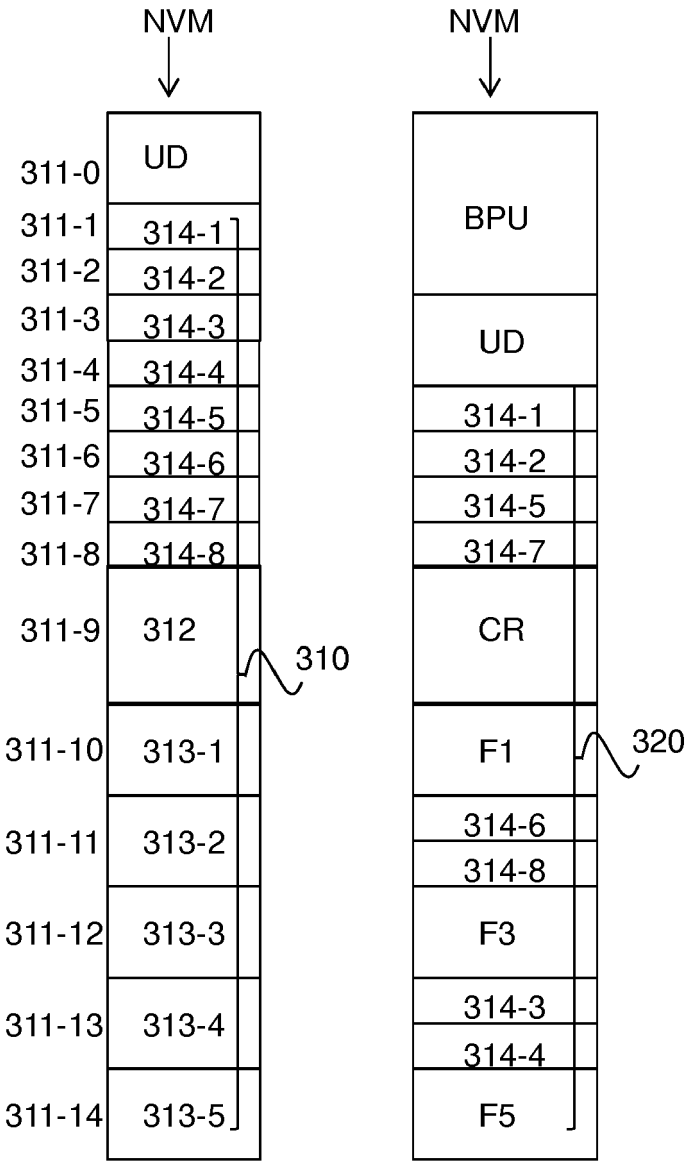
NVM
UD
311-0
311-1
311-2
311-3
311-4
311-5
311-6
311-7
311-8
311-9
311-10
311-11
311-12
311-13
311-14
314-1
314-2
314-3
314-4
314-5
314-6
314-7
314-8
312
313-1
313-2
313-3
313-4
313-5
310
NVM
BPU
UD
314-1
314-2
314-5
314-7
CR
F1
314-6
314-8
F3
314-3
314-4
F5
320

Fig. 9a Fig. 9b

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 18 30 6551

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EP 3 067 795 A1 (GEMALTO SA [FR]) 14 September 2016 (2016-09-14) * paragraph [0002] - paragraph [0066] * * figures 1-6 * * claims 1-10 *	1-14	INV. G06F8/54 G06F8/41 G06F9/445 G06F21/57
A	WO 2017/151546 A1 (650 IND INC [US]) 8 September 2017 (2017-09-08) * paragraph [0029] - paragraph [0047] * * figures 3-4 *	1-14	
A	WO 2014/044403 A2 (GIESECKE & DEVRIENT GMBH [DE]) 27 March 2014 (2014-03-27) * page 1 - page 3 * * page 9 - page 18 *	1-14	

TECHNICAL FIELDS SEARCHED (IPC)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	16 May 2019	Rackl, Günther

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 18 30 6551

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

16-05-2019

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 3067795 A1	14-09-2016	EP 3067795 A1	14-09-2016
		EP 3268858 A1	17-01-2018
		WO 2016142381 A1	15-09-2016
WO 2017151546 A1	08-09-2017	EP 3423938 A1	09-01-2019
		US 2019073230 A1	07-03-2019
		WO 2017151546 A1	08-09-2017
WO 2014044403 A2	27-03-2014	EP 2898413 A2	29-07-2015
		WO 2014044403 A2	27-03-2014

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82